Paper

# A Queue Monitoring System in OpenFlow Software Defined Networks

Shiva Rowshanrad, Sahar Namvarasl, and Manijeh Keshtgari

*Computer Engineering and IT Department, Shiraz University of Technology, Shiraz, Iran*

**Abstract—Real-time traffic characteristic is different and it is very sensitive to delay. To meet traffic specifications in real time, monitoring systems are used as an important part of networking. Many monitoring systems are deployed to have an update view of the network QoS parameters and performance. Most of these systems are implemented to measure QoS parameters in links. Here, in this paper, a system for monitoring queues in each link by means of Software Defined Networks is proposed. The monitoring system is implemented by extending Floodlight controller, which uses OpenFlow as southbound protocol. The controller has a centralized view of the network. By the help of OpenFlow it also can provide flow level statistics. Using these advantages, the proposed system can monitor delay and available bandwidth of a queue on a link or path. Despite of monitoring systems in traditional networks, the proposed monitoring system makes a low overhead in network thanks to OpenFlow protocol messages. It is also integrated into the network controller, which enables QoS and traffic engineering applications to use the system's reports for automatic traffic management and QoS setup. The experimental results show a 99% accuracy of the proposed system for monitoring of both bandwidth and delay.**

***Keywords—Floodlight, OpenFlow, queue monitoring, Software Defined Networks.***

## 1. Introduction

In today's networks, there are many kinds of traffic, such as video streaming, video conferencing, VoIP, FTP, etc. Each of these traffics has different QoS requirements. VoIP and video conferencing need less than a 150 ms delay and less than 1% packet loss [1]. Therefore, these data types need different traffic engineering (TE) to transmit efficiently. Queuing disciplines are common examples of TEs. However, in order to have a complete traffic management a queuing monitoring system is required, which can provide a real-time report about QoS parameters of each queue on a requested path or link.

Many monitoring systems have been proposed in traditional networks, but usually they consume too many resources such as bandwidth and computational power, they may need additional hardware and are not very accurate nor flexible. Also some of these methods may not work properly on heavy load [2], [3].

Recently many have taking advantage of Software Defined Networks (SDNs) and OpenFlow protocol to create monitoring systems. In SDN, control plane is separated from data plane and placed into a centralized server named controller. The controller communicates with network forwarders using an open interface such as OpenFlow [4]. Each forwarder keeps track of counters related to packets and flows. The controller can be aware of these counters by polling forwarders using OpenFlow statistic messages. The counters are per table, per flow, per port, per queue, including received packet/byte, transmitted packet/bytes, their duration, number of received/transmitted drops, etc. [5].

In this paper a system for monitoring delay and available bandwidth of queues, is proposed. It provides TE for QoS management.

The remainder of this paper is organized as follows. In Section 2, related works in SDNs and traditional networks are described. Section 3 introduces different parts of the proposed monitoring system. In Section 4, the experimental tests and their results are presented. Finally, Section 5 concludes the paper.

## 2. Related Works

One of the earliest tools for network monitoring is Simple Network Management Protocol (SNMP). It uses port counters across every switch to estimate links load. Although SNMP is widely used in many monitoring devices in traditional networks, it has some drawbacks. First, it may result high CPU overhead. Moreover, SNMP is unable to collect flow-level statistics and measuring metrics such as loss and delay. It also requires additional infrastructure [6], [7].

The sFlow [8] and NetFlow [9] are two flow-based monitoring systems, which use packet sampling. They both use agents at switches and routers to sample packets and collect statistics. sFlow agents can push the information to a centralized collector, while NetFlow agents would be polled by the collector.

Recently many monitoring/measurement systems for SDNs are proposed. IBM had leveraged sFlow sampling and implemented OpenSample over Floodlight controller [10]. Using packet sampling from flows, which have sequence numbers (e.g. TCP), by centralized collector, OpenSample can have a fast detection of elephant flows and link utiliza-

Table 1
Monitoring systems comparison

| Monitoring system | Network | Parameters | Method | Implementation | Strengths and weaknesses |
|---|---|---|---|---|---|
| SNMP | Traditional | Bandwidth/link utilization | Poll port counters which gather information about packets | SNMP manager and agents | • single parameter,<br>• high CPU utilization,<br>• needs additional infrastructure |
| sFlow | | | Packet sampling agents push information to a centralized collector | sFlow agents and collector | • single parameter,<br>• can be used in SDN but not appropriate,<br>• flow-based measurement,<br>• needs additional infrastructure |
| NetFlow | | | Polling packet sampling agents | NetFlow agents and collector | • single parameter,<br>• can be used in SDN but not appropriate,<br>• high CPU usage,<br>• Flow-based measurement,<br>• needs additional infrastructure |
| OpenSample | SDN | Flow/link utilization | Pocket sampling agents push information to a centralized collector | Floodlight controller | • single parameter,<br>• without end device modification,<br>• high accuracy,<br>• low latency |
| PayLess | | Bandwidth/link utilization | Polling OpenFlow (OF) statistics from switches | Flodlight controller | • single parameter,<br>• trade-off between accuracy and overhead in different polling intervals |
| FlowSense | | Per flow link utilization | Push based | Not described | • single parameter,<br>• high accuracy,<br>• low overhead |
| Phemius *et al.* [2] | | Link latency | Polling OF messages, use of probe packets | Floodlight controller | • single parameter,<br>• high accuracy,<br>• low overhead |
| OpenTM | | Bandwidth/link utilization | Polling based (5 different methods) | NOX controller | • single parameter,<br>• high accuracy,<br>• high overhead on edge switches |
| OpenNetMon | | Per flow packet loss, delay and throughput | Polling OF messages, use of probe packets | POX controller | • trade-off between accuracy and overhead |

tion. The throughput of OpenSample can be up to 150% over sFlow in some cases.

In [11] a monitoring system named PayLess is implemented over Floodlight [12]. PayLess collects statistics from switches by polling them. The applications on top of the controller can request the desired QoS metrics, which can be extracted from these statistics, using RESTful API. As a use case, PayLess was evaluated for link utilization information. The results show that PayLess can collect more accurate statistics than FlowSence [13], which estimates link utilization by analyzing control messages sent from switches to the controller instead of polling switches. In addition, PayLess messaging overhead is 50% of overhead in an equivalent method using polling statistics.

In [2] a link latency monitor over Floodlight controller is proposed. The method for measuring latency is based on sending an Ethernet frame, with an unknown Ethernet-type value, over the link from the controller and measuring the time until the packet comes back. The difference of this time from half of the Round Trip Time (RTT) of statistic messages between edge switches of the link and controller would be the latency value of that link. The overhead of this method is 81% less than ping utility while its accuracy is 99.25% comparing to ping.

In [14] OpenTM is presented. OpenTM is a traffic matrix estimator, implemented using NOX controller [15]. It presents the traffic load between each pair of switches in an OpenFlow network by polling statistics. Due to packet

loss, the statistics of different switches in path would have different results. For this reason, five design methods were compared to each other: polling the last switch, polling switches uniformly at random, round robin querying, none uniform random polling, polling the least loaded switch. The evaluation results show that polling the last switch gives the most accurate values. However, it makes a high load on edge switches.

OpenNetMon [3] is another SDN monitoring system. It is implemented over POX controller and monitors per-flow metrics such as throughput, delay and packet loss. The flow throughput is measured using statistics related to amount of bytes sent in a flow and its duration. The required statistics are achieved by polling the last switch, while in case of packet loss measurement. The statistics are polled from the first and last switch of the path. The packet loss value is calculated from subtract of switches packet counters. An equivalent method as in [2] is used for delay monitoring by means of probe packets. The results of evaluation show that the proposed method for monitoring throughput is quite accurate, while methods for packet loss and delay measurements are not. These inaccuracies caused by lack of synchronization between measurement setups and software fluctuations. Table 1 shows the comparison between mentioned monitoring systems.

## 3. Proposed Monitoring System

All the systems, which have been proposed in SDNs were for monitoring links. Here, in this paper a monitoring system, which can monitor queues in each link or path in terms of delay and available bandwidth is proposed. The authors extended Floodlight controller, which uses OpenFlow protocol, for implementing this system.

The key parameter of QoS is bandwidth. If the bandwidth were insufficient for transmitting traffics, loss and delay would be also unfavorable. For measuring the used bandwidth of each queue on each link, there is a need to poll the "transmitted bytes" counter of the queue from the switch at the head of the link. It is required to keep track of the last polling time and counter to calculate the used bandwidth of queue. Then the bandwidth would be calculated as:

$$BWq = \frac{TB_n - TB_{n-1}}{t_n - t_{n-1}}, \tag{1}$$

where $BWq$ stands for used bandwidth of the desired queue, $TB$ is the transmitted bytes and $t$ is the polling time in seconds. The free capacity can be calculated by subtracting $BWq$ from the base bandwidth of the queue.

Delay is another important QoS parameter, which also can have a huge effect on packet loss. Different applications have different delay requirement, so it is important to determine, which queue can best satisfy which application at the moment.

The delay of each link can be calculated by sending a packet with arbitrary Ethernet-type value from the controller to the switch located at the head of the link. Then the switch sends this packet to the next switch at the other side of the link. As there is no entry in the flow table, matching this Ethernet-type value, the second switch sends the packet back to the controller. So the controller knows the times of sending and receiving the packet, hence it knows the duration.

For measuring delay of a queue, the packet should be queued. It means beside the output action, an enqueue action must be set for the packet. In this case, the time between sending and receiving the packet consists of queuing delay and propagation delay. Multi-arbitrary Ethernet-type values can be used to determine between the probe packets related to each queue.

For having the exact delay of queue on a link between two switches, it is necessary to subtract the measured time from half of RTT's between the switches and the controller. This RTT can be measured using Statistics_Request and Statistics_Reply messages. Equation (2) shows the delay of a queue on a specific link:

$$\text{Delay}_q = T_c - \frac{\text{RTT}_{s1}}{2} - \frac{\text{RTT}_{s2}}{2}, \tag{2}$$

where $T_c$ is the duration of sending and receiving the probe packet by controller.

## 4. Experimental Results and Performance Evaluation

The monitoring module was implemented by extending Floodlight controller. OpenFlow was used as the southbound protocol and OpenVswitch [16] was installed on Ubuntu 14.04 for creating OpenFlow switches. A linear topology with three switches and four hosts running on Intel Core i5/i7 CPUs PC with Windows 8/Ubuntu 14.04 (as shown in Fig. 1) was used.

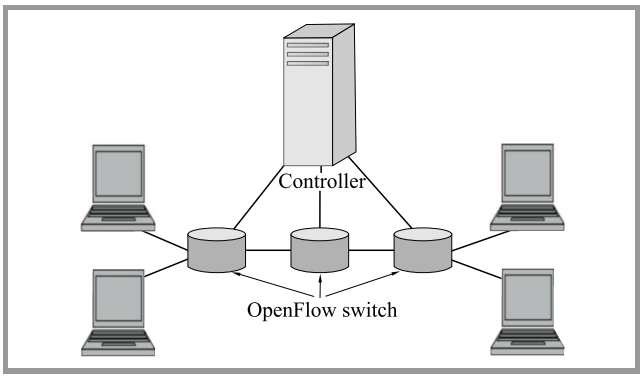


***Fig. 1.*** Performance evaluation topology.

For testing the proposed monitoring system, two queues with maximum bandwidth of 30 Mb/s were created on

each link, and then special TCP and UDP flows such as video streaming, video conferencing, etc., were allocated to each, by means of floodlight RESTful API. We also tried to generate different TCP/UDP flows using traffic generators in every host, flowing to the opposite one. The flows were increased in time to fulfill the queues' bandwidth in about 15 s. The switches polling interval of proposed system is set to 1 s. The bandwidth was also checked in every second by JPERF tool. Some special flows were monitored by Wireshark to be analyzed for delay. As the proposed system can measure the latency of queues in links between switches, Wireshark application also set to monitor the packets on edge switches to have accurate measurements.

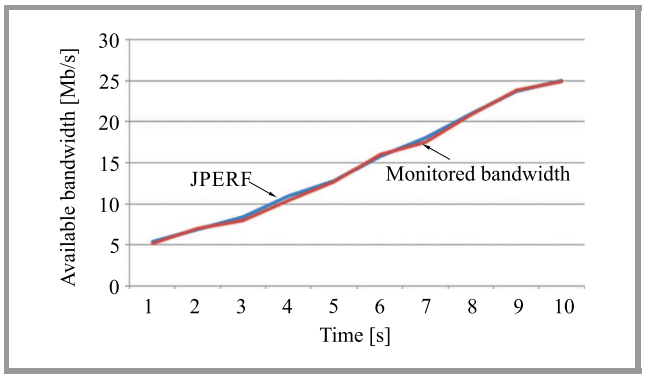


***Fig. 2.*** Comparing monitored bandwidth by JPERF and proposed monitoring method. (See color pictures online at www.nit.eu/publications/journal-jtit)

Figure 2 shows the monitored bandwidth in first 10 s of the test. According to analysis in Table 2, the proposed method results are as the same as JPERF results because the 95% confidence interval of system averages includes zero. From the averages, we can say that the proposed system accuracy compared to JPERF is 99%. One of the advantages of proposed system over JPERF is that the system can monitor bandwidth even if the queue is full. JPERF may face difficulties in high-load networks from when there is a little bandwidth left, and crashed in last seconds of test.

Table 2
Monitored bandwidth analysis

| JPERF average bandwidth [Mb/s] | Monitored average bandwidth [Mb/s] | 95% confidence interval of averages difference |
|---|---|---|
| 7.01 | 7.07 | (–0.03, 0.33) |

For delay measurement performance evaluation, the delays of both queues were monitored with monitoring module. Also the average delay of some special flow was calculated from the monitored results of Wireshark.
The q0 is the lowest priority queue while the q1 has higher priority. Figure 3 shows the monitored delay in

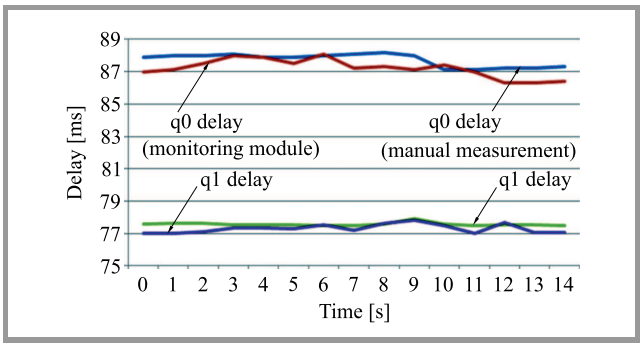


***Fig. 3.*** Comparing monitored delay of queues.

Table 3
Delay analysis of monitored queues

| Delay in 15 s | Average of monitoring module measurement in 15 s | Average of manual measurement in 15 s | 95% confidence interval of averages difference |
|---|---|---|---|
| q0 | 87.21 | 87.73 | (0.28, 0.78) |
| q1 | 77.30 | 77.57 | (0.13, 0.41) |

each second for these queues. Although the calculated delay values, monitored delay values and their averages are close to each other, the confidence intervals in Table 3 do not include zero. This means that the systems do not have the exactly the same results. This difference can be due to the difference between delays of each two different flows in the queue. The delay values of flows in a queue differ in a short range because of their difference in packet size, processing time, etc. Therefore, the monitored delay can be considered as an estimated delay of desired queue in desired path with a high accuracy about 99%.

## 5. Conclusion

In this paper, a monitoring system for measuring queue QoS parameters such as available bandwidth and delay is proposed. The proposed monitoring system is the first system for monitoring queue parameters for SDNs. It is implemented over Floodlight controller and uses OpenFlow statistic messages and probe packets to measure the mentioned parameters. Use of OpenFlow protocol messages gives the advantage of monitoring network with low network overhead. Integrating the system as a software module makes it independent of using network devices' resources. It also is a cheaper system comparing to traditional monitoring systems, as it doesn't need extra infrastructure. The performance evaluation of the system shows an accuracy of 99% for measuring both available bandwidth and delay of a queue in a desired path or link. The next step is to use this monitoring system for networks, which uses queuing as their TE method, for optimizing network's performance automatically.

# References


[1] T. Szigeti and C. Hattingh, *Quality of Service Design Overview*. San Jose, CA: Cisco Press, 2004, pp. 15–16.

[2] K. Phemius and M. Bouet," Monitoring latency with OpenFlow", in *Proc. 9th Int. Conf. on Netw. and Service Manag. CNSM 2013*, Zürich, Switzerland, 2013, pp. 122–125 (doi:10.1109/CNSM.2013.6727820).

[3] N. L. M. van Adrichem, C. Doerr, and F. A. Kuipers, "OpenNetMon: Network monitoring in OpenFlow software-defined networks", in *Proc. IEEE/IFIP Network Operations and Management Symp. NOMS 2014*, Kraków, Poland, 2014, pp. 1–8 (doi:10.1109/NOMS.2014.6838228).

[4] S. Rowshanrad, S. Namvarasl, A. Abdi, M. Hajizadeh, and M. Keshtgary, "A survey on SDN, the future of networking", *J. of Adv. Comp. Sci. & Technol.*, vol. 3, no. 2, pp. 232-248, 2014 (doi:10.14419/jacst.v3i2.3754).

[5] OpenFlow Switch Consortium, OpenFlow Switch Specification Version 1.0.0., 2009 [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

[6] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Boston: Addison-Wesley Longman Publ. Co., 1998.

[7] J. Case, M. Fedor, M. Schoffstall, and C. Davin, "A simple network management protocol (SNMP)", RFC 1098, Network Information Center, SRI International, Menlo Park, CA, USA, 1989 [Online]. Available: http://archive.openflow.org/documents/openflow-spec-v1.0.0.pdf

[8] P. Phaal, "sFlow Specification Version 5", July 2004 [Online]. Available: http://sflow.org/sflow_version_5.txt (accessed: 25 May 2015).

[9] B. Claise, "Cisco systems NetFlow services export version 9", 2004 [Online]. Available: http://tools.ietf.org/html/rfc3954.html (accessed: 25 May 2015).

[10] J. Suh, T. Kwon, C. Dixon, W. Felter, and J. Carter, "OpenSample: A low-latency, sampling-based measurement platform for SDN", in *Proc. IEEE 34th Int. Conf. Distrib. Comput. Syst. ICDCS 2014*, Madrid, Spain, 2014, pp. 228–237 (doi:10.1109/ICDCS.2014.31).

[11] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "PayLess: A low cost network monitoring framework for software defined networks", in *Proc. IEEE/IFIP Netw. Operat. and Manag. Symp. NOMS 2014*, Kraków, Poland, 2014 (doi:10.1109/NOMS.2014.6838227).

[12] Floodlight OpenFlow Controller [Online]. Available: http://www.projectfloodlight.org/floodlight/ (accessed: 25 May 2015).

[13] C. Yu, C. Lumezanu, Y. Zhang, V. Singh, G. Jiang, and H. V. Madhyastha, "FlowSense: Monitoring network utilization with zero measurement cost" in Proc. 14th Int. Conf. on Passive and Active Measurement PAM'13, Hong Kong, China, 2013, pp. 31-41 (doi:10.1007/978-3-642-36516-4_4).

[14] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "OpenTM: Traffic matrix estimator for OpenFlow networks", in Proc. 11th Int. Conf. on Passive and Active Measurement PAM'10, Zurich, Switzerland, 2010, pp. 201–210 (doi:10.1007/978-3-642-12334-4_21).

[15] NOX [Online]. Available: http://www.noxrepo.org/nox/about-nox/ (accessed: 25 May 2015).

[16] OpenVSwitch [Online]. Available: http://openvswitch.org (accessed: 25 May 2015).



**Shiva Rowshanrad** received her M.Sc. degree in Information Technology Engineering (Computer networks field) from Shiraz University of Technology (SUTECH), Shiraz, Iran. Her main research interest is Software Defined Networking. Her other research interests are Named Data Networking, Wireless Sensor Networks and multimedia.
E-mail: shiva.rrad@gmail.com
Computer Engineering and IT Department
Shiraz University of Technology
Shiraz, Iran



**Sahar Namvarasl** received her B.Sc. in Information Technology from Shiraz University of technology (SUTECH), Shiraz, Iran. She is currently working toward M.Sc. degree at SUTECH, majoring in computer networks. Her research interests are in the area of Software Defined Networks, virtualization and cloud computing.
E-mail: sahar.namvarasl@gmail.com
Computer Engineering and IT Department
Shiraz University of Technology
Shiraz, Iran



**Manijeh Keshtgari** is a faculty member of Department of Computer Engineering and IT, Shiraz University of Technology, Shiraz, Iran. She received her M.Sc. degree in Electrical and Computer Engineering from Colorado State University, CSU, Fort Collins, USA in 1993 and her Ph.D. degree in Computer Engineering from Sharif University of Technology in 2005. Her research interests include MANET, Wireless Sensor Networks and GSM security issues.
E-mail: keshtgari@sutech.ac.ir
Computer Engineering and IT Department
Shiraz University of Technology
Shiraz, Iran